Amendments to the Specification:

Please amend paragraph **[0009]** as indicated to add a space between "Fig." and "4":

[000105] Fig. 4 is a block diagram illustrating at least one embodiment of a format for spill and fill micro-ops generated by at least one embodiment of a register stack engine.

Please amend **Table 2** in paragraph **[00059]** as indicates to delete the space between "Load" and "8" in line 9 of **Table 2**:

Table 2

1.	void doOneFill () {
2.	INT64 x;
3.	bool grflag;
4.	i = (i-1) % M;
5.	**bspload%i% -= (8*M);**
6.	grflag = EXTRACT (bspload%i%, 8, 3)!=63;
7.	if (grflag) loadreg -= 1;
8.	if (grflag) { // load a general register
9.	**~~Load 8~~ Load8 GR[loadreg].value = [bspload%i%];**
10.	**GR[loadreg].nat = rnatextract (RNAT, EXTRACT (bspload%i%, 8, 3));**
11.	} else { // load the RNAT register
12.	**Load8 RNAT = [bspload%i%];**
13.	};
14.	};

Please amend paragraph **[00080]** as indicated:

[00080] In addition, at block 604, a micro-op may be generated to decrement the value in the architecturally-visible bspload application register. An illustrative example of a bspload pre-decrement micro-op that may be generated at block 608 is set forth at line 5 of **Table 2 ~~3~~**, above. Processing then proceeds to block 606.

Please amend paragraph **[000106]** as indicated:

[000106] From block 718, processing then proceeds to block 720, where one or more double-wide spill micro-ops is generated to perform the double-wide spill to the backing store 151. An example of a micro-op that may be generated at block 720 is set forth at line 15 of **Table 3**. Because the spill (Store) micro-op indicates a double-wide load operation, the value of bspstore is incremented in order to account for the additional backing store entry that has been processed during the current iteration. Accordingly, the Store16 micro-operation increments the bspstore address. For at least one embodiment, this increment is performed by zero-ing out bit three of the address held in bspstore. The sample micro-op set forth at line ~~16~~ 15 of **Table 3** indicates that this may be accomplished by performing a Boolean AND of the bspstore address and the complement of the hexadecimal value "8" to mask out bit 3 to a value of zero. Accordingly, on the first and second pass of the method 700 for "Spill series A", internal instructions are generated to collect the first and second halves of the temporary value, tmpreg. On the second iteration of the method 700, the low and high halves of tmpreg are stored to the backing store in a single cycle, effectively writing two entries into the backing store 151 during a single cycle.

Please amend paragraph **[000111]** as indicated:

[000111] From blocks 714, 720 and 722, processing proceeds to block 724. At block 724, variables are post-incremented. For at least one embodiment, both internal and external variables are incremented. Line 20 of **Table 3** illustrates a micro-op that may be generated at block 724 in order to post-increment the architecturally-visible bspstore application register. In addition, line 19 of ~~19of~~ **Table 3** illustrate an example instruction that may cause the internal variable storereg to be incremented if grflag is true; a true value for grflag indicates that a general register (rather than the RNAT) was spilled during the current iteration of the method

700. Otherwise, if the RNAT was spilled (i.e., grflag = false) then storereg is not incremented.

From block 724, processing for the method 700 ends at block 726.